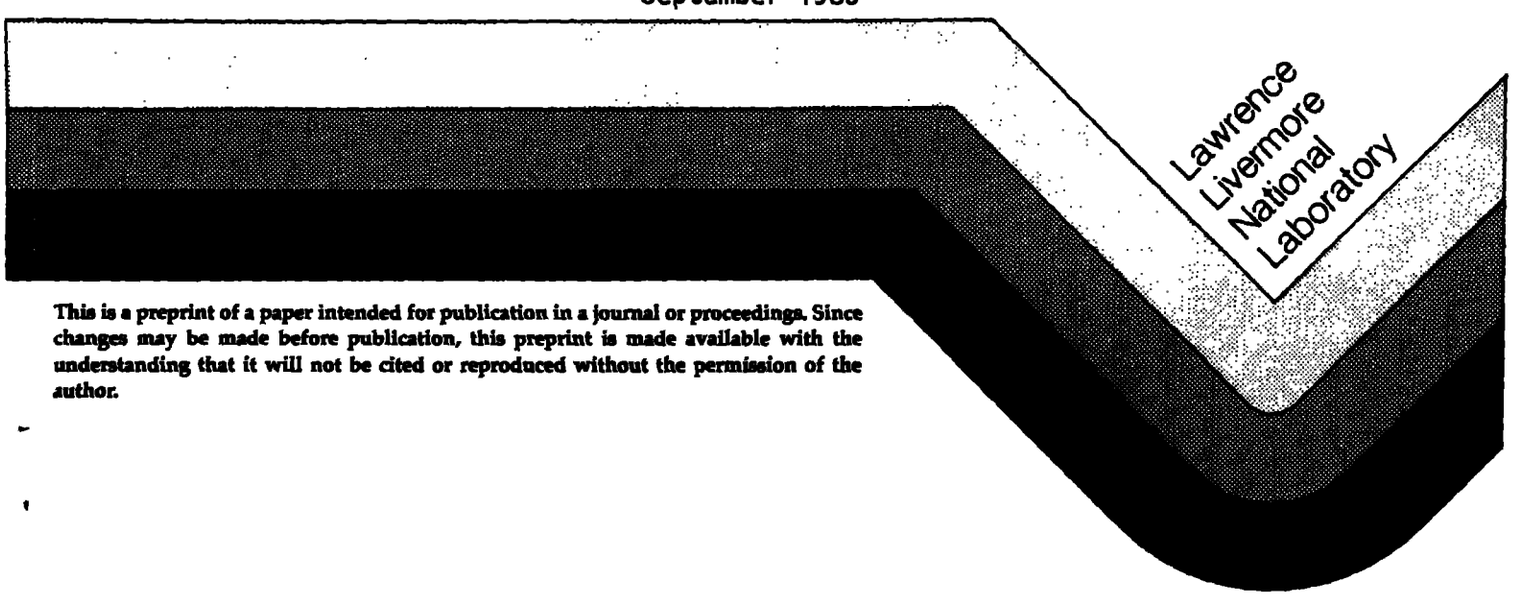
UCRL- 93357
PREPRINT

The Supercomputer - A Friendly Tool on the Scientists' Desk

D. Fuss

This paper was prepared for submittal to the
SHARE European Association Meeting in Zuerich,
Switzerland, September 23-27, 1985

September 1985



THE SUPERCOMPUTER - A FRIENDLY TOOL ON THE SCIENTIST'S DESK*

D. Fuss, National Magnetic Fusion Energy Computer Center, Lawrence Livermore National Laboratory,P.O. Box 5509, Livermore, California, USA

1 Introduction

Supercomputers are being used in a variety of relatively new applications - airplane design, weather forecasting, oil reclamation, energy research, and structural analysis, to name a few. Thus, the supercomputer is rapidly moving from an exotic tool used by a few experts to the indispensable tool of many researchers. This raises questions about how to make supercomputers easier to use, how to provide access to supercomputers for remote users, and how to maximize the efficiency of both users and supercomputers. This paper will discuss some of the trade-offs and provide some points of view, gained through experience, on these issues.

2 Are Supercomputers Domesticated?

When R.M. Pirzig wrote the following statement in "Zen and the Art of Motorcycle Maintenance," he was referring to motorcycles, of course. But the analogy is an apt one for supercomputers, too.

> The material objects...can't be right or wrong. They don't have any ethical codes to follow except those people give them. The test of the machine is the satisfaction it gives you. There isn't any other test. If the machine produces tranquility, it's right. If it disturbs you, it's wrong until either the machine or your mind is changed.

This paper will attempt to do both--change the machine as well as the mind.

3 What Is a Supercomputer Anyway?

A gigaflop machine?
A computer that costs more than $10 million?
A computer with 512 megabytes of memory?

For the purposes of this discussion, a supercomputer is a general purpose computer that can execute more operations per second and address more storage than other computers.

*Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

The following table lists some current supercomputers and their execution rate in Millions of Floating Point Operations per Second (MFLOPS), as measured by the 14 Livermore Fortran kernels.

Kernel	Description	Cray X-MP (1 CPU) (Opt 1985)	Cray-2 (1 CPU) (Opt 1985)	CYBER 205	Fijitsu VP200	Hitachi S810/20	NEC SX2
1	Hydro Excerpt	125.0	166.7	79.1	326.0	228.0	601.8
2	Inner Product	83.3	69.0	88.0	178.0	239.0	408.8
3	Inner Product	83.3	90.9	88.0	331.0	211.9	528.5
4	Banded Linear Equations	42.5	32.9	12.2	88.0	59.2	118.8
5	Tridiagonal (above)	9.6	9.3	5.4	10.0	5.4	13.5
6	Tridiagonal (below)	7.4	8.3	6.5	10.0	4.6	12.7
7	Equation of State	147.7	213.0	51.0	325.0	232.7	800.0
8	Part. Diff. Eq. Int.	75.8	80.0	15.7	90.0	48.8	157.6
9	Integral Predictors	141.7	170.0	47.3	257.0	207.6	558.1
10	Difference Predictors	31.0	56.2	23.5	84.0	49.0	125.0
11	First Sum	3.3	3.5	7.6	5.0	9.8	23.8
12	First Difference	41.7	52.6	86.2	114.0	93.0	235.5
13	Particle Pusher	3.9	4.9	2.0	6.0	4.2	8.1
14	1-D Particle Pusher	6.1	8.6	4.3	13.0	8.5	22.3
	Mean	57.5	69.0	36.9	131.2	100.1	258.2
	Harmonic Mean	12.7	14.6	8.4	19.8	14.7	35.1

These figures were obtained from DATAMATION (9/84), from NMFECC's monthly publication, the BUFFER (6/85), and from data supplied by NEC Corporation of Japan. The source listing of the kernels is published in COMPUTER (10/84). Note that the Cray X-MP and Cray-2 are really four processor systems, so the harmonic mean could easily be close to four times what it is if the kernels were multiprocessed.

4 Why Bother With a-Supercomputer?

- Supercomputers are the general purpose mainframes of the future. Even data processing computers are old supercomputers in architecture.

- It's much more economical to operate one 100 MFLOP machine than five 20 MFLOP machines. Staffing, maintenance, floor space, networking, and faster turnaround all argue in favor of a single, powerful mainframe. Reliability is so good that usually redundancy is not necessary.

- Historically, scientists who use computers have restricted their numerical simulations to an average execution time of about ten hours. This constraint reflects the scientist's need to make daily progress. Thus, the amount of complexity incorporated in models is scaled to the computer's ability to produce results in a given amount of time. The capability of a supercomputer allows for much greater complexity to be treated in a given amount of time.

$$\text{Response Time}\left(\frac{\text{Time}}{\text{Problem}}\right) = \frac{\text{Complexity}}{\text{Execution Rate}}\ \frac{\left(\frac{\text{Results}}{\text{Problem}}\right)}{\left(\frac{\text{Results}}{\text{Time}}\right)}$$

The execution rate can increase if any one of a computer system's three elements are improved--system hardware, system software, and application software. As one begins to understand a new hardware architecture, software improvements can continue to take place over the lifetime of the supercomputer. Execution rate, then, is a function of hardware and the sophistication (maturity) of the system and application software.

Complexity in modeling is primarily a function of dimensionality, resolution, and physics.

- Dimensionality--The real world exists in three space dimensions plus time. If computational models could reflect the real world exactly, they would address all four of these dimensions as well as other parameters that are equivalent to additional dimensions. With current supercomputers, it is possible to treat two space dimensions and time for some problem types, three space dimensions for others, and three space dimensions plus time for a very limited set of problems.

- Resolution--Every region of space contains infinitely many points. Thus, the first step in modeling any natural phenomenon is to approximate the space with a finite set of zones, each of which requires a number of calculations. Increasing the number of zones means one can determine more completely and accurately what is happening in any environment, but the computational time grows very rapidly. For example, in a two-dimensional time-dependent model, the running time grows proportional to the third power of the increase in resolution; increasing the number of zones by just a factor of 2 would increase the time to complete the problem by a factor of 8.

- Physics--All computational models dealing with the frontiers of science and technology make simplifying assumptions about the laws of physics in order to prevent the calculation from running too long. In some models, including just one additional physical effect can increase running time by a factor of 10.

Although dimensionality, resolution and physics each have powerful effects on execution time by themselves, the overall impact is derived from combinations of these effects. Thus the question becomes: Given the opportunity to execute your computational model on a supercomputer, would you put up with some extra work or inconvenience to get much better (or even brand new) results from your model? If the answer is "Yes, depending on how much extra work or how much inconvenience," then we can turn from

changing your mind to changing the machine and to the subjects of supercomputer access and software.

5 What is Required for Remote Access to Supercomputers?

The acquisition of a supercomputer frequently leads to the question of how to provide remote access to it. Usually one single group of users at one geographic location does not have the computational requirements to justify the acquisition. If one can provide remote access to a whole company or to several research institutions at many locations, one can usually come up with a total requirement to justify a supercomputer. It is usually a more cost-effective solution than providing smaller computers at many sites. This is all based on the assumption that one can provide reasonable remote access at a reasonable cost. Let's address that issue.

There are various views on what is required for remote access to a supercomputer, and certainly a case can be made for all of these views, but let us begin by satisfying the average user.

- The average user can be served with an intelligent desktop terminal (IBM PC or equivalent microprocessor) that can be used to communicate with the supercomputer over a 1200 baud dial-up telephone line. This "less than $2000 front-end" provides:

  - Access to a timesharing system on the supercomputer (to be discussed in detail later).
  - Distributed screen editing. A distributed screen editor combines the intrinsic responsiveness of a personal computer with the power of a supercomputer. It literally distributes the work between the supercomputer and the PC. The supercomputer, for example, can supply information upon request, update the file, and perform global searches and replacements. The PC will display a "window" into the file, interpret commands, update whatever is displayed, and forward the changes to the host supercomputer.
  - Graphics capability using, for example, Tektronix emulation. This is slow via 1200 baud, but once a picture is in the memory of the PC, one should be able to zoom in or pan out. Good software can allow the graph to be transmitted into the PC while the user is working on something else in another window of the PC.
  - For voluminous output one uses the postal service.

- Depending on the nature of the work and the number of people accessing the supercomputer from a single site, a minicomputer directly connected to the supercomputer with a dedicated 9600 baud line can be a cost-effective solution. This "remote user service station" can

  - serve as a terminal concentrator to replace the 1200 baud dial-up connections previously mentioned.
  - support a printer for good quality and reasonable quantity graphic and alphanumeric hardcopy output.
  - provide substantially better (faster and more flexible) graphic capability than one can get with a PC. Graphic data for raster displays or printers can typically be compressed to less than 10%

of the expanded data for transmission over a data link. One needs, however, a reasonable computer to uncompress data at the remote site.

- With a substantial number of users at one geographic location, one can probably justify a reasonably high (56 or 112 k-bit/sec) link between the supercomputer and a small computer like a VAX. The VAX or equivalent can

  - serve as the "remote user service station" with faster file transfer.
  - provide some local compute capability.
  - serve as a gateway to a local area network or other "remote user service stations."

This solution can be viewed as "front-ending" the supercomputer, but that is not the case. Front-ends require very high bandwidth, much higher than noted here. Broad (T-1) bandwidth is extremely expensive and cannot realistically be provided to a dispersed geographic user population. Additionally, broad bandwidths may not be necessary for most supercomputer applications.

6 Should One Timeshare on a Supercomputer?

In my discussion of remote access to a supercomputer, I assumed that a timesharing system would be available on it. This is not a unanimously accepted idea, but it should be. The more users there are using a supercomputer to aid them in their work, the more important it becomes to increase the efficiency of the user, perhaps at the expense of optimizing the hardware. A timesharing system has these obvious advantages:

- A complete Batch environment is naturally available.

- The need for a front-end disappears. The normal justification for a front-end is that one needs mature (stable and capability-rich) software that is usually only available on older mainframes.

- Symbolic and dynamic debugging can be provided. In most cases, this capability is impossible to provide on a front-end (the exception is if an emulator of the supercomputer exists that is exactly instruction and register compatible). Symbolic and dynamic debugging are absolutely essential on today's supercomputers; vector processing is hard enough to debug, but multiprocessing is even more difficult.

- Results can be reviewed immediately to increase the number of turnarounds and to avoid unnecessary calculations. One should be able to stop a job at any time, symbolically look at what is going on, and continue the job or even change some variables and continue the calculation in a new direction.

Some experts say that the disadvantage of timesharing on a supercomputer is that it wastes valuable resources, namely cycles for number crunching. Although it is true that a timesharing system requires more CPU cycles and I/O than a Batch system, one must weigh all of the factors. Communication with a front-end also requires CPU cycles and I/O, does not provide important functionality (symbolic and dynamic debugging), and contains additional costs in terms of networking and acquiring the front-end. One

cannot accurately measure the increase in efficiency that a timesharing system provides for its user community, but one can measure what affect timesharing has on the total resources of a supercomputer (CPU, memory, and I/O). Some measured statistics are attached.

7 So, How Do You Get Good Software on a Brand New Supercomputer?

History has shown that supercomputer hardware is invariably ready for use long before any reasonable software. But the unique performance of a new type of supercomputer is most valuable at the time of its initial delivery--that is the time when one group of scientists may have a computational edge over scientists who do not have access to such a new supercomputer. Consequently, it is important to learn how to produce a complete, functional software environment that is ready when the new computer becomes available. Equally important, the software environment should be one that is familiar so that users can move their applications quickly. Even though a new supercomputer may be architecturally quite different from a previous one, advance preparation of the software is possible.

Here are some rules that we have found advantageous to follow:

- Write all software--the system, utilities, and libraries--in ONE higher level language; say Fortran (written in itself) with extensions that make it possible to do systems development (bit and byte manipulation, structuring of words into bit fields, etc.). This way, one compiler effort assists with the software development process and also meets daily applications requirements. Assembly language code should be kept to a minimum. Limit assembly language code to those areas that significantly benefit and even those sections of code should have Fortran equivalents, whenever possible, for portability.

- Develop libraries in layers with the bottom layer containing a set of basic functions that are machine and system dependent. The other (higher-level) layers will contain libraries that are machine and system independent. Users who want to get to the system call level will use these lower-level libraries; normal users will use the higher-level libraries. These comments apply particularly to I/O and graphic libraries. For example, if a user wants to create a disk file and read or write to that file, the higher-level library calls will do that very easily. If that same user wants to "double buffer" disk I/O, he may want to guarantee that I/O is performed on two distinct disks on separate controllers by specifically requesting physical disks for his files; this would be accomplished by calling lower-level library routines. With this layering approach, only a few basic libraries have to be prepared for the new architecture.

- Concentrate the software effort when preparing for a new supercomputer on the MOVE; this is not the time to simultaneously develop new system capabilities that cannot be checked out in a normal production environment. The MOVE should center only on adapting to the new architecture.

• Build a software simulator of the new architecture on an existing mainframe in such a way that mature software (debugged libraries, utilities, system) can be used. Then, with a cross compiler, assembler, loader and binary library builder for the new architecture, all software can be checked out one piece at a time. For example, someone checking out a utility will cross compile the utility, but will link with mature libraries to load the code, and will then execute the code on the simulator. Someone checking out a library can proceed independently to cross compile the library, and then execute converted utilities that have been tested on the simulator with the mature libraries.

• With the cross compiler, assembler, loader and binary library builder in place, and a simulator on an existing mainframe, one can proceed with the following :

- Convert all assembly language code to the new assembly language.
- Convert and test all libraries (I/O, graphic, math, etc.)
- Modify the symbolic and dynamic debugger to recognize the new hardware structure, new state save package, new disassembler, and new symbol tables.
- Convert and test all utilities and sub-systems (e.g., file transport, file manipulation routines, output routines, accounting and administrative control routines, and the Batch system)

• Simultaneously with these efforts, the system must be modified to adapt to the new architecture. This may include a number of the following:

- Changing system tables, possibly because of a larger memory address field.
- Re-coding assembly coded sections.
- Writing new I/O drivers for new hardware interfaces.
- Providing new diagnostic tools (history buffers) for understanding the characteristics of the new hardware and for isolating problems.

Once the software is operational on the new hardware, THEN one can spend time on optimizing the compiler for the new architecture and other issues. But with the basic software in place, users can begin to execute their codes and become familiar with the characteristics of the new architecture. Optimization of the software will continue over the lifetime of the machine.

The procedures and philosophy described here were used at the National Magnetic Fusion Energy Computer Center to prepare a complete software environment for the Serial #1 Cray-2. A multiprocessing system (CTSS), which resides on the Cray-1 and the Cray X/MP at the NMFECC, was modified to execute on the Cray-2. (The Cray-2 is a 4-processor, 500 megabyte memory system that differs from the Cray X/MP in instruction set, register organization, memory access, I/O interface, and timing characteristics.) Users began executing applications on the Cray-2 within a week after the hardware system was installed at the NMFECC. The development of this initial software environment took 20 person-years.

In conclusion, today's supercomputers can and should be a friendly tool on the scientist's desk. This can be accomplished with as little as a PC with a 1200 baud modem, a good timesharing system on a supercomputer, and some distributed editing software. The benefit of providing such a capability is that the scientist can then perform more accurate, realistic calculations of physical phenomena without having to become a computer wizzard.

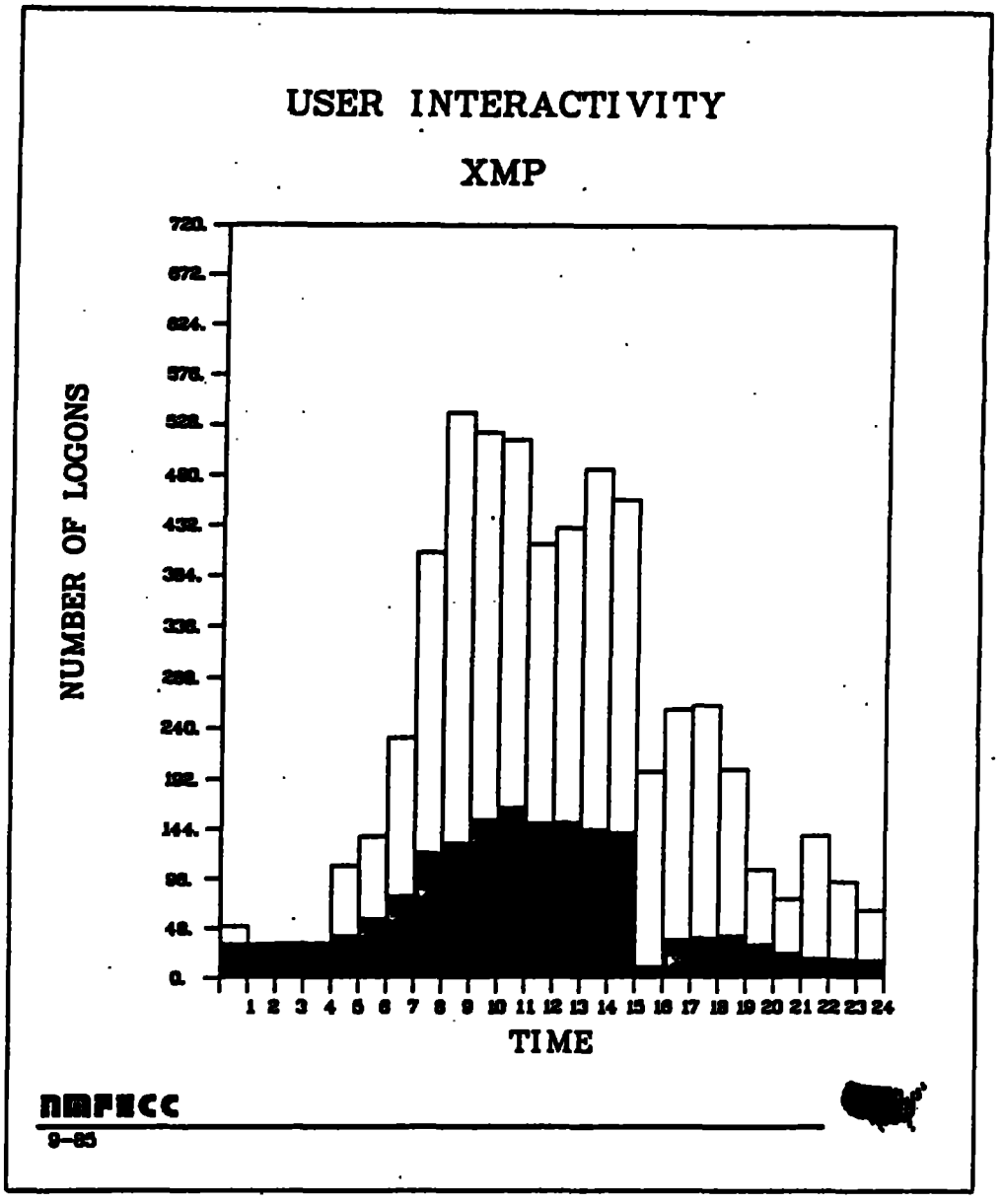


Figure 1. The clear bars represent the total number of users logged onto the CRAY X-MP during each hour of a typical 24-hour day. The black bars represent the number of users logged in at the end of each hour.

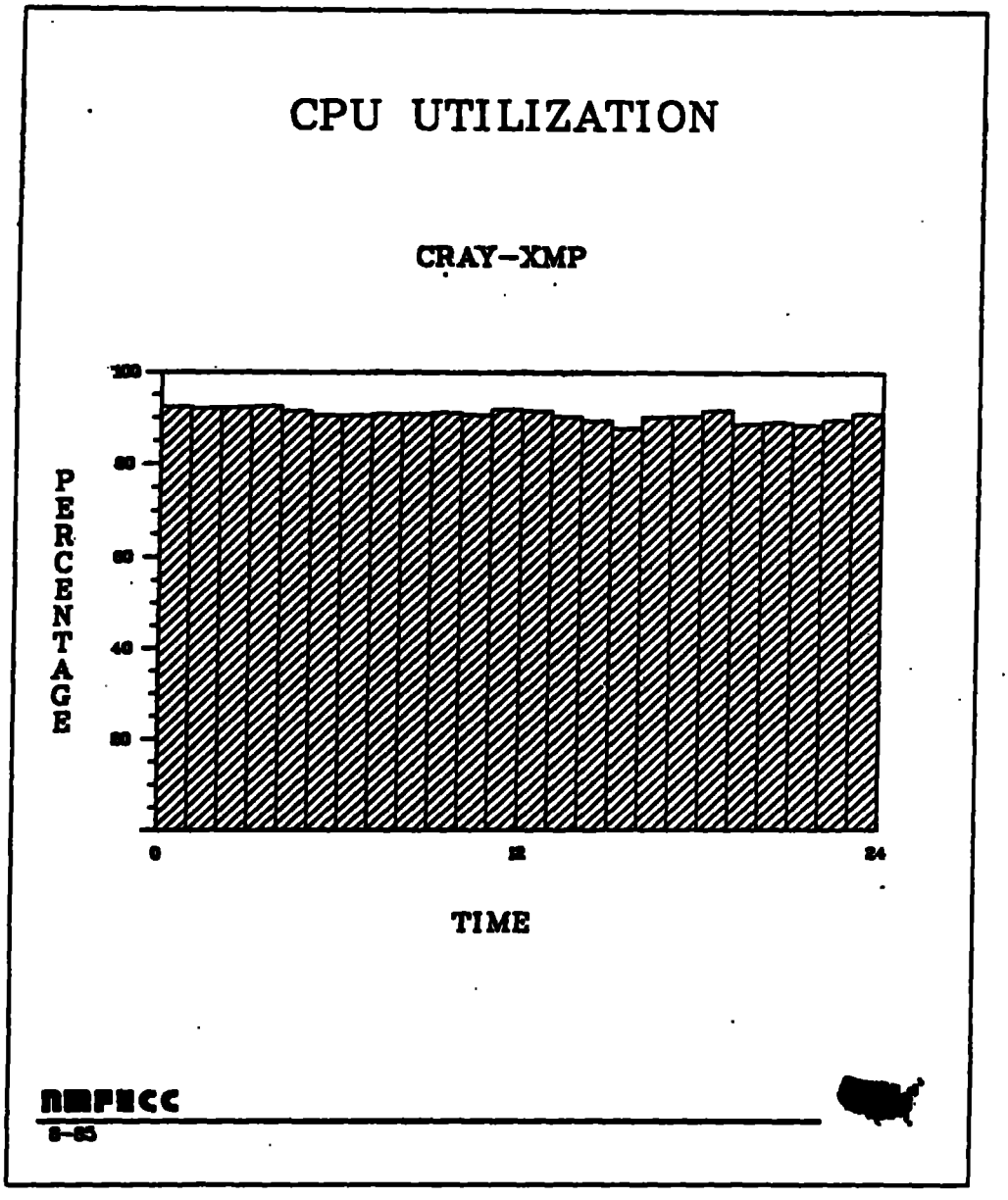


Figure 2. The crosshatched bars show the percent of CPU delivered to users during each hour of a typical day on a 2-processor CRAY X-MP. The difference between 100% and the crosshatched bars is CPU used by the system and/or idle time.

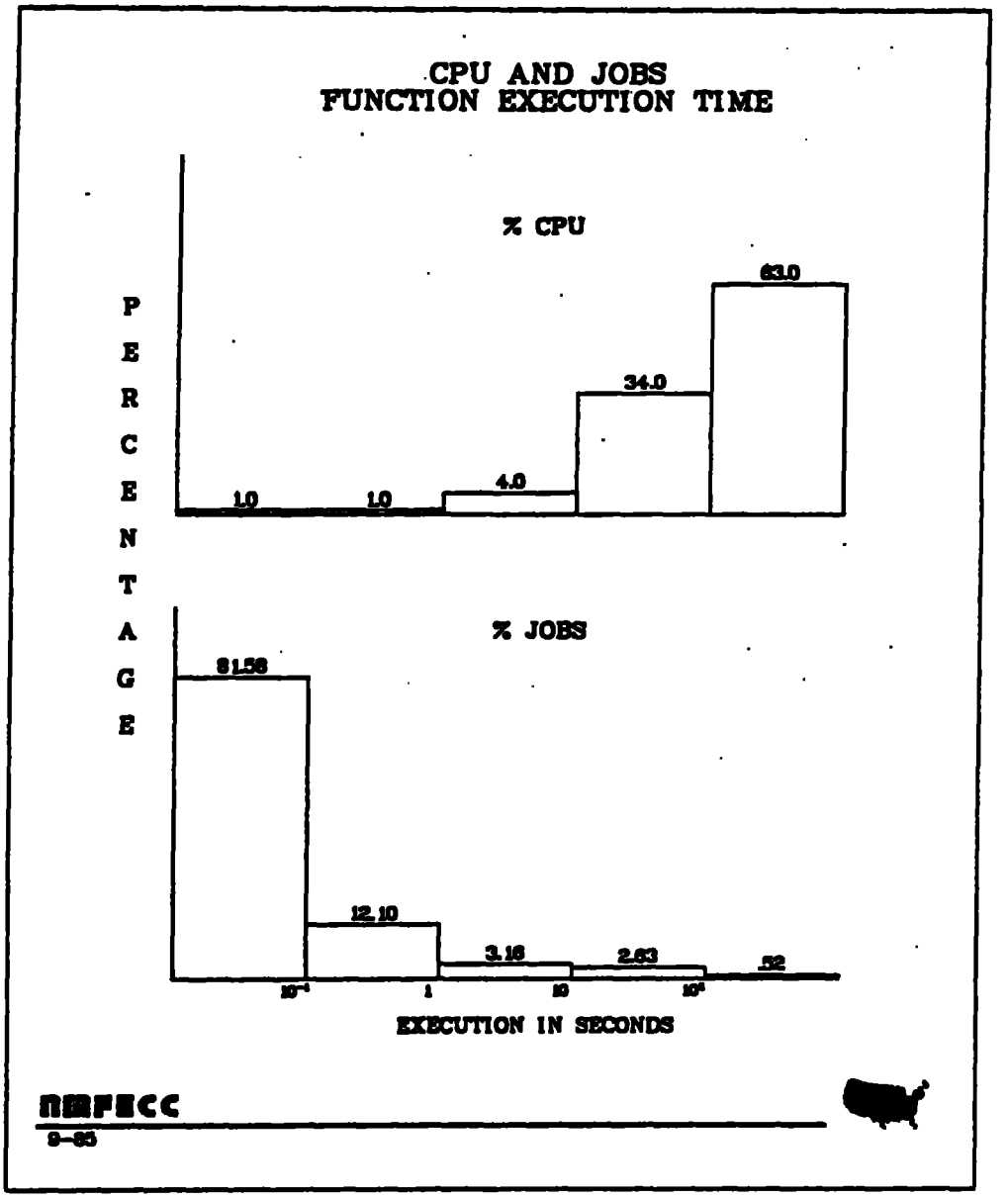


Figure 3. The bottom graph plots the percent of the total number of jobs against execution time. The top graph plots how much CPU was used by the jobs in each time interval.